Electronic Version
Stylesheet Version v1.1.1

Description

System Providing Methodology for the Restoration of Original Media Quality in Messaging Environments

COPYRIGHT STATEMENT

BACKGROUND OF INVENTION

[0002] 1. Field of the Invention

[0003] The present invention relates generally to preservation of media (e.g., images, audio, video, documents, and the like) quality, and relates more particularly to restoring media quality in messaging environments.

[0004] 2. Description of the Background Art

[0005] MMS (Multimedia Messaging Service) is a universally accepted standard that lets users of MMS supportive mobile phones send and receive messages with formatted text, graphics, photographs, audio, and video clips. Video sequences, audio clips, and high-quality images can be downloaded to the phone from WAP (Wireless Application Protocol) sites, transferred to the phone via an attached accessory (e.g., a digital camera), or received in an MMS message. MMS messages can be sent either to another MMS-enabled mobile phone or to an e-mail address. Photos, sound clips, and video clips can also be stored in the phone for later use. MMS supports standard image formats such as GIF and JPEG, video formats such as MPEG 4, and audio formats such as MP3 and MIDI. Multimedia messaging requires high transmission speeds, which can be provided by GPRS and 3G. To support the MMS technology, existing GSM or CMDA networks need an MMS-C (Multimedia Messaging Service Center).

[0006] These multimedia messaging environments provide a message generation service that allows a variety of message elements to be sent to a user. These message elements can contain text, animations, photographs, and

sounds. In the future they may contain streaming audio and video. Users can compose their own messages, receive rich content messages from content providers, and forward them onto their own contacts. Examples include:

[0007] Taking a snapshot via a camera phone and sending to a friend

[0008] Composing your own animated picture messages and sending to friends

[0009] Sending audio files

[0010] Sending pictures & audio files with simultaneous playback

[0011] Receiving a picture postcard annotated with text and/or an audio clip

[0012] Further, the user can also send messages from a variety of devices, including mobile phones, PDAs, Palm Pilots, or PCs. MMS makes it possible for mobile users to send these multimedia messages from MMS-enabled handsets to other mobile users and to e-mail users. It also makes it possible for mobile users to receive multimedia messages from other mobile users, e-mail users, and from multimedia-enabled applications.

[0013] In messaging systems like MMS, there is no central repository for storing media items or objects that are shared

among subscribers. This is in contrast to many picture mail services, where a central switching server also stores the media objects (e.g., components in user-composed messages) centrally and persistently. In MMS, messages are instead created, and objects are sourced, at individual devices or terminals connected to the system without any central storage mechanism. Although a central switch in the form of a server may be employed to facilitate message delivery, the central switch or server does not attempt to operate as a central repository for storing or retaining media objects (unlike, say, picture mail services). MMS messages are composed using a standards-compliant messaging format and sent through a switch (e.g., MMSC for a MMS system) to one or more destinations (i.e., target devices or terminals). This is a "stored forward" approach, such that a given a message and its associated objects are stored on a temporary basis. Although the media objects may pass through one or more servers, any storage is at most transient in MMS systems; instead, the message and its media objects are stored just long enough to achieve delivery of the message. Once a given message has reached its target destination, the message's media objects will typically have been transcoded or oth-

erwise decimated (e.g., convert media format, size, or the like), in an effort to optimize the rendering of the media objects for the target destination. Upon successful delivery of the message, the message itself and all of its associated media objects are discarded from temporary storage (e.g., at the central switch), or moved to off-line archival storage.

[0014] Given the simplicity of the MMS model where the central switch is only a transient switch, any high-quality media (e.g., audio, video, still images, etc.) that is destined for target devices that have less than full capability for rendering (e.g., less-capable display) will undergo quality degradation. In particular, the MMS-C switch will take the original high-quality media and create new lower-quality versions. Consider, for example, an original high-quality image. Here, the switch will often need to create a lower resolution image, in order to provide an appropriately-formatted image for the destination device. Apart from capability constraints of a given target device, bandwidth constraints may also mandate that media objects be converted into lower-quality copies (to conserve bandwidth). The destination device may use received media objects for composing other messages (e.g., forward or reply mes-

sages), but because of the foregoing degradation process the reused media objects become progressively poorer in quality. The scenario is similar to faxing and re-faxing a document. By the time the document has been faxed three or four times, the quality has degraded to the point where the fax copy is no longer usable. In the particular context of MMS, since there is no central repository for storing media objects, once a given media object has been degraded, the high quality of the original is not preserved because the association with the original media object has itself not been preserved. Overall, this leads to loss of quality and loss of flexibility for media items that are used in multimedia messaging systems.

[0015] In messaging systems of this type, as soon as the original media object (that is part of a message) is transcoded or otherwise converted for delivery to another mobile or terrestrial terminals, the original media quality is lost to the system (i.e., for subsequent use). To the point, the quality of delivered original media objects is forever lost for subsequent users as the system does not retain the quality in any way, shape, or form. Subsequent users of these delivered media items or objects (for new messages) are forever constrained to work with subsequent lower-quality

versions.

SUMMARY OF INVENTION

[0016] A system providing methodology for the restoration of original media quality in messaging environments is described. In one embodiment, for example, deployed in a messaging system, a method of the present invention is described for restoring media items to original quality, the method comprises steps of: upon receipt of a message containing an original media item that is new, storing the original media item in a repository; generating an identifier for identifying the original media item stored in the repository; replacing the original media item in the message with a substitute copy that includes the identifier; and upon future encounter of a particular media item having the identifier, restoring the particular media item to original quality using the identifier.

[0017] In another embodiment, for example, a system of the present invention is described for restoring media items to original quality, the system comprises: a messaging system capable of transmitting multimedia messages; a repository for storing the original media item upon receipt of a message containing an original media item that is new; a module for generating an identifier for identifying

the original media item stored in the repository; a module for replacing the original media item in the message with a substitute copy that includes the identifier; and a module for restoring the particular media item to original quality using the identifier.

Brief Description of Drawings

[0018] Fig. 1 is a block diagram of a computer system in which software-implemented processes of the present invention may be embodied.

[0019] Fig. 2 is a block diagram of a software system for controlling the operation of the computer system.

[0020] Fig. 3A is a block diagram that illustrates a high-level view of a system providing restoration of original media quality in a messaging environment.

[0021] Fig. 3B is a block diagram that provides a more detailed view of the media original reference identification module.

[0022] Figs. 4A-B comprise a flowchart illustrating a method of the present invention for handling new or original incoming media.

[0023] Figs. 5A-B comprise a flowchart illustrating a method of the present invention for processing incoming media objects that contain ORI (object reference identifier) information.

[0024] Fig. 6 is a simplified block diagram of the JPEG image file format, for illustrating the embedding of ORI information in a media object.

DETAILED DESCRIPTION

GLOSSARY

[0025] The following definitions are offered for purposes of illustration, not limitation, in order to assist with understanding the discussion that follows.

[0026] 3G: This is an ITU specification for the third generation (analog cellular was the first generation, digital PCS the second) of mobile communications technology. 3G promises increased bandwidth, up to 384 Kbps when a device is stationary or moving at pedestrian speed, 128 Kbps in a car, and 2 Mbps in fixed applications. 3G will work over wireless air interfaces such as GSM, TDMA, and CDMA.

[0027] Digital watermark: A pattern of bits inserted into a digital image, audio file, or video file that includes identification information, such as the file's copyright information (author, rights, etc.). The name comes from the faintly visible watermarks imprinted on stationery that identify the manufacturer of the stationery. The primary purpose

of digital watermarks is to provide copyright protection for intellectual property that is in digital format.

[0028] DRM: Short for digital rights management, a system for protecting the copyrights of data circulated via the Internet by enabling secure distribution and/or disabling illegal distribution of the data. Typically, a DRM system protects intellectual property by either encrypting the data so that it can only be accessed by authorized users or marking the content with a digital watermark or similar method so that the content can not be freely distributed. See, e.g., IETF Internet DRM Working Group (currently at www.idrm.org).

[0029] GPRS: Short for General Packet Radio Service, a standard for wireless communications which runs at speeds up to 115 kilobits per second, compared with current GSM (Global System for Mobile Communications) systems' 9.6 kilobits. GPRS, which supports a wide range of bandwidths, is an efficient use of limited bandwidth and is particularly suited for sending and receiving small bursts of data, such as e-mail and Web browsing, as well as large volumes of data.

[0030] GSM: Short for Global System for Mobile Communications, one of the leading digital cellular systems. GSM uses nar-

row-band TDMA, which allows eight simultaneous calls on the same radio frequency.

[0031] Media: Refers broadly to any content that may be represented in digital form, such as audio, video, images, documents, or the like.

[0032] MM1 (also, MMS Library): A library or protocol that allows a mobile client to send and receive messages containing a variety of media types, e.g., text, voice, image and video clips using Multimedia Messaging Services (MMS).

[0033] MM7: This protocol consists of abstract request and response messages. As distinct from the earlier interfaces between VASP (Value Added Service Platform) applications and MMS Relay/Server, MM7 is a common vendor-independent protocol. MM7 protocol is based on the concept of Web Services and uses SOAP and HTTP for communication. The multimedia messages are sent to the MMS Relay/Server with HTTP POST method. The body of the post contains XML data about the delivery and the multimedia message as a MIME-multi-part attachment.

[0034] MMS: Short for Multimedia Message Service, a store-and-forward method of transmitting graphics, video clips, sound files and short text messages over wireless networks using the WAP protocol. Carriers deploy special

servers, dubbed MMS Centers (MMSCs) to implement the offerings on their systems. MMS also supports e-mail addressing, so the device can send e-mails directly to an e-mail address. Mobile and other similar user devices may compose e-mail-like messages that contain one or more media items (e.g., images, audio, video, text, and the like), that may be sent to another subscriber. The most common use of MMS is for communication between mobile phones.

[0035] TCP/IP: Stands for Transmission Control Protocol/Internet Protocol, the suite of communications protocols used to connect hosts on the Internet. TCP/IP uses several protocols, the two main ones being TCP and IP. TCP/IP is built into the UNIX operating system and is used by the Internet, making it the de facto standard for transmitting data over networks. For an introduction to TCP/IP, see e.g., RFC 1180: A TCP/IP Tutorial, the disclosure of which is hereby incorporated by reference. A copy of RFC 1180 is available on the Internet (e.g., currently from ietf.org).

[0036] WAP: Short for the Wireless Application Protocol, a secure specification that allows users to access information instantly via handheld wireless devices such as mobile phones, pagers, two-way radios, smart phones, and com-

municators.

INTRODUCTION

[0037] Referring to the figures, exemplary embodiments of the invention will now be described. The following description will focus on the presently preferred embodiment of the present invention, which is implemented in desktop and/or server software (e.g., driver, application, or the like) operating in an Internet-connected environment running under an operating system, such as the Microsoft Windows operating system. The present invention, however, is not limited to any one particular application or any particular environment. Instead, those skilled in the art will find that the system and methods of the present invention may be advantageously embodied on a variety of different platforms, including Macintosh, Linux, BeOS, Solaris, UNIX, NextStep, FreeBSD, and the like. Therefore, the description of the exemplary embodiments that follows is for purposes of illustration and not limitation. The exemplary embodiments are primarily described with reference to block diagrams or flowcharts. As to the flowcharts, each block within the flowcharts represents both a method step and an apparatus element for performing the method step. Depending upon the implementation, the

corresponding apparatus element may be configured in hardware, software, firmware or combinations thereof.

COMPUTER-BASED IMPLEMENTATION

[0038] *Basic system hardware (e.g., for desktop and server computers)*

[0039] The present invention may be implemented on a conventional or general-purpose computer system, such as an IBM-compatible personal computer (PC) or server computer. Fig. 1 is a very general block diagram of an IBM-compatible system 100. As shown, system 100 comprises a central processing unit(s) (CPU) or processor(s) 101 coupled to a random-access memory (RAM) 102, a read-only memory (ROM) 103, a keyboard 106, a printer 107, a pointing device 108, a display or video adapter 104 connected to a display device 105, a removable (mass) storage device 115 (e.g., floppy disk, CD-ROM, CD-R, CD-RW, DVD, or the like), a fixed (mass) storage device 116 (e.g., hard disk), a communication (COMM) port(s) or interface(s) 110, a modem 112, and a network interface card (NIC) or controller 111 (e.g., Ethernet). Although not shown separately, a real time system clock is included with the system 100, in a conventional manner.

[0040] CPU 101 comprises a processor of the Intel Pentium family

of microprocessors. However, any other suitable processor may be utilized for implementing the present invention. The CPU 101 communicates with other components of the system via a bi-directional system bus (including any necessary input/output (I/O) controller circuitry and other "glue" logic). The bus, which includes address lines for addressing system memory, provides data transfer between and among the various components. Description of Pentium-class microprocessors and their instruction set, bus architecture, and control lines is available from Intel Corporation of Santa Clara, CA. Random-access memory 102 serves as the working memory for the CPU 101. In a typical configuration, RAM of sixty-four megabytes or more is employed. More or less memory may be used without departing from the scope of the present invention. The read-only memory (ROM) 103 contains the basic input/output system code (BIOS) -- a set of low-level routines in the ROM that application programs and the operating systems can use to interact with the hardware, including reading characters from the keyboard, outputting characters to printers, and so forth.

[0041] Mass storage devices 115, 116 provide persistent storage on fixed and removable media, such as magnetic, optical

or magnetic-optical storage systems, flash memory, or any other available mass storage technology. The mass storage may be shared on a network, or it may be a dedicated mass storage. As shown in Fig. 1, fixed storage 116 stores a body of program and data for directing operation of the computer system, including an operating system, user application programs, driver and other support files, as well as other data files of all sorts. Typically, the fixed storage 116 serves as the main hard disk for the system.

[0042] In basic operation, program logic (including that which implements methodology of the present invention described below) is loaded from the removable storage 115 or fixed storage 116 into the main (RAM) memory 102, for execution by the CPU 101. During operation of the program logic, the system 100 accepts user input from a keyboard 106 and pointing device 108, as well as speech-based input from a voice recognition system (not shown). The keyboard 106 permits selection of application programs, entry of keyboard-based input or data, and selection and manipulation of individual data objects displayed on the screen or display device 105. Likewise, the pointing device 108, such as a mouse, track ball, pen device, or the like, permits selection and manipulation of objects on the

display device. In this manner, these input devices support manual user input for any process running on the system.

[0043] The computer system 100 displays text and/or graphic images and other data on the display device 105. The video adapter 104, which is interposed between the display 105 and the system's bus, drives the display device 105. The video adapter 104, which includes video memory accessible to the CPU 101, provides circuitry that converts pixel data stored in the video memory to a raster signal suitable for use by a cathode ray tube (CRT) raster or liquid crystal display (LCD) monitor. A hard copy of the displayed information, or other information within the system 100, may be obtained from the printer 107, or other output device. Printer 107 may include, for instance, an HP LaserJet printer (available from Hewlett Packard of Palo Alto, CA), for creating hard copy images of output of the system.

[0044] The system itself communicates with other devices (e.g., other computers) via the network interface card (NIC) 111 connected to a network (e.g., Ethernet network, Bluetooth wireless network, or the like), and/or modem 112 (e.g., 56K baud, ISDN, DSL, or cable modem), examples of

which are available from 3Com of Santa Clara, CA. The system 100 may also communicate with local occasionally-connected devices (e.g., serial cable-linked devices) via the communication (COMM) interface 110, which may include a RS-232 serial port, a Universal Serial Bus (USB) interface, or the like. Devices that will be commonly connected locally to the interface 110 include laptop computers, handheld organizers, digital cameras, and the like.

[0045] IBM-compatible personal computers and server computers are available from a variety of vendors. Representative vendors include Dell Computers of Round Rock, TX, Hewlett-Packard of Palo Alto, CA, and IBM of Armonk, NY. Other suitable computers include Apple-compatible computers (e.g., Macintosh), which are available from Apple Computer of Cupertino, CA, and Sun Solaris workstations, which are available from Sun Microsystems of Mountain View, CA.

[0046] *Basic system software*

[0047] Illustrated in Fig. 2, a computer software system 200 is provided for directing the operation of the computer system 100. Software system 200, which is stored in system memory (RAM) 102 and on fixed storage (e.g., hard disk) 116, includes a kernel or operating system (OS) 210. The

OS 210 manages low-level aspects of computer operation, including managing execution of processes, memory allocation, file input and output (I/O), and device I/O. One or more application programs, such as client application software or "programs" 201 (e.g., 201a, 201b, 201c, 201d) may be "loaded" (i.e., transferred from fixed storage 116 into memory 102) for execution by the system 100. The applications or other software intended for use on the computer system 100 may also be stored as a set of downloadable computer-executable instructions, for example, for downloading and installation from an Internet location (e.g., Web server).

[0048] System 200 includes a graphical user interface (GUI) 215, for receiving user commands and data in a graphical (e.g., "point-and-click") fashion. These inputs, in turn, may be acted upon by the system 100 in accordance with instructions from operating system 210, and/or client application module(s) 201. The GUI 215 also serves to display the results of operation from the OS 210 and application(s) 201, whereupon the user may supply additional inputs or terminate the session. Typically, the OS 210 operates in conjunction with device drivers 220 (e.g., "Winsock" driver -- Windows' implementation of a TCP/IP stack) and the

system BIOS microcode 230 (i.e., ROM-based microcode), particularly when interfacing with peripheral devices. OS 210 can be provided by a conventional operating system, such as Microsoft Windows 9x, Microsoft Windows NT, Microsoft Windows 2000, or Microsoft Windows XP, all available from Microsoft Corporation of Redmond, WA. Alternatively, OS 210 can also be an alternative operating system, such as the previously mentioned operating systems.

[0049] The above-described computer hardware and software are presented for purposes of illustrating the basic underlying desktop and server computer components that may be employed for implementing the present invention. For purposes of discussion, the following description will present examples of a preferred embodiment in which it will be assumed that there exist one or more "servers" (e.g., message switch server and media object reference identifier module server) that communicate with one or more "clients" (e.g., wireless terminals, handheld devices, desktop computers, and/or the like). The present invention, however, is not limited to any particular environment or device configuration. In particular, a client/server distinction is not necessary to the invention, but is used to provide a framework for discussion. Instead, the present

invention may be implemented in any type of system architecture or processing environment capable of supporting the methodologies of the present invention presented in detail below.

OVERVIEW OF RESTORATION OF ORIGINAL MEDIA QUALITY IN MESSAGING ENVIRONMENTS

[0050] From a high-level perspective, the present invention seeks to use one or more already-available data communication channels that exist within the encoding of various media objects, so that a given original media object may be referenced or indexed for future use. As messages containing a given media object are passed through a switching center, the individual switching center may obtain a reference for the media object -- the original reference identifier -- using the methodology of the present invention, and embed or associate that reference with the media object. In one embodiment, the switching center may retain both the original reference identifier and an original copy of the media object so that any subsequent decimated copy that is used may be re-associated with the original copy, thereby allowing the system to in effect create a higher-quality, first generation copy for the target device. Each decimated copy retains enough information (by the

embedded or associated original reference identifier) to allow the copy to be re-associated with the original media object.

[0051] In the currently preferred embodiment, the present invention is implemented in a message switch-based environment. A "switch" is usually embodied as software and hardware components that exist at the carrier's infrastructure (e.g., at a wireless cellular network). In an MMS-based switch, for example, a server component is used to listen for the receipt of messages from mobile terminals (e.g., data processing-enabled cellular phones) on well-known ports, using typical Internet-like connectivity mechanisms. As these systems are usually subscription based, the switch typically includes a module for identifying/validating/billing the subscriber terminals that connect. The server includes functionality for decoding the requests from the mobile terminals that comprise the transmission of one or more multimedia objects, as part of a standards-defined multimedia message. The switch includes switching capability in the form of target/destination resolution, using a variety of address resolution schemes. As part of that target/destination resolution, the switch includes functionality (or default rules)

that allows it to transmit a media object in its original quality (uncommon) or to decimate the object, as required for the particular capabilities of a given destination (e.g., for display on a small screen of a hand-held device). After a message is delivered to a given destination, the message no longer exists, except for potential archiving purposes. However, the message, including all of it multimedia context components, is at that point no longer available for all practical purposes to subscriber terminals.

SYSTEM COMPONENTS

[0052] Fig. 3A is a block diagram that illustrates a high-level view of a system 300 providing restoration of original media quality in a messaging environment. As shown, the system includes a message switch 320 connected to mobile terminals 310, 330, which typically comprise "smart" cellular phones (e.g., with microprocessor and display screen) and the like. In operation, the message switch 320 communicates by sending messages using the MM1 protocol. The message switch 320, which may include the Nokia MMS-C message switch or similar product, includes a media conversion facility. Although mobile terminals 310, 330 are shown as separate source and destination terminals for purposes of illustration, typically any given

terminal may serve as a source for messages (source mobile terminals 310) or a destination for messages (destination mobile terminals 330). Further, the present invention does not require that these source and destination terminals in fact be mobile. Instead, they need only be clients capable of sending or receiving multimedia messages. In a typical embodiment, these clients will be mobile devices of various capabilities.

[0053] As also shown, the system 300 includes a media original reference identification module 350 of the present invention, which in turn has access to an original media repository 351. In the currently preferred embodiment, the module 350 is embodied as a server that communicates with the message switch 320 via an application protocol (e.g., MM7 protocol). This protocol includes the capability to bear arbitrary application commands. This protocol is employed to define a set of application services that allow the module 350 to communicate with the message switch 320 in a manner that allows original reference identifiers to be provided for media objects passing through the system. The message switch 320, depending on architecture, may include a plug-in module or driver facilitating communication with the module 350. In the currently pre-

ferred embodiment, an object reference identifier (ORI) service driver 321 is employed for this purpose.

[0054] The module 350 may be viewed as an add-on module for the message switch 320. Message traffic passing through the message switch 320 may be trapped or "hooked", by the above mentioned plug-in or driver. Upon trapping a given message, the message switch 320 provides the module 350 with a message identifier (subscriber information, etc.) and all media objects of the message. In response, the module 350 may assign an original reference identifier (ORI) for the message, as well as each media object of the message. The ORI-identifiable media may now be stored in the original media repository 351, for future use. The repository 351 may be implemented in a conventional database, such as Oracle 9 (available from Oracle Corp. of Redwood Shores, CA). In the currently preferred embodiment, the repository 351 employs an aging mechanism so that stale or obsolete items are eventually "aged out" (removed) from storage.

[0055] In basic operation, the module 350 embeds the ORI information within each of the media objects. The particular technique used to embed the ORI information within a given media object will depend on the underlying media

type (e.g., JPEG, MP3, etc.) for the object. The modified media object (i.e., with the embedded ORI information) is returned to the message switch 320 as a substitute for the original object. This substitute, which is an ORI-embellished copy that retains the original's quality, may now be transformed or converted (as required, if any) and sent to the target destination. The ORI-embedded information is immune to any transformation or conversion, and thus will be preserved in the copy that is sent to the target destination. The result is that decimated media contains ORI-embedded information that allows identification of an original copy.

[0056] Because the terminals all communicate through the message switch 320, the ORI service driver 321 constantly monitors incoming media for ORI-embedded information. Accordingly, the message switch 320 may re-associate an incoming decimated media object (coming in from a terminal) with the original media object, if that original had been seen previously by the message switch. In basic operation at this point, the message switch 320 hands the incoming message to the ORI service driver 321. The re-association of a decimated media object with its original may be done (i.e., by the module 350 and repository 351)

in a manner that is transparent to the message switch 320.

[0057] Fig. 3B is a block diagram that provides a more detailed view of the media original reference identification module 350. As shown, the module 350 includes ORI embedding unit 361, ORI assign unit 363, ORI scanning unit 365, and services layer 367. As previously mentioned, the module 350 connects to a repository 351 that stores media objects. The repository 351 may be optionally indexed (e.g., by ORI value), as indicated by index 352, to facilitate object access and retrieval.

[0058] The ORI embedding unit 361 includes program logic for embedding an ORI value in a given media object. Accordingly, the unit 361 includes knowledge about different media types, so that it may embed an ORI value in a media object of a particular type in a manner that will allow the embedded value to survive any subsequent transformation (e.g., such as applied by the message switch 320). The ORI assign unit 363 includes program logic for formulating an ORI reference value, in instances where a newly-encountered media object is presented at the message switch 320 (i.e., presentation of new media). The ORI scanning unit 365 includes program logic to scan incom-

ing media objects for any embedded ORI information. In instances where decimated media with embedded ORI information is encountered, the scanning unit 365 may re-associate the decimated media with the ORI-maintained original (in the repository 351). The services layer 367 provides application protocol services or interface(s) that allow the module 350 to communicate with one or more switches. The interface(s) includes command support for the aforementioned embedding, assigning, and scanning features, including forwarding media objects to the module 350 for storage and returning substitute (reconstituted) media objects to the message switch 320.

[0059] This approach solves the problem of the continued destruction of the fidelity/quality of media objects that pass through existing multimedia messaging systems. Further, because a copy of the original media object may be located, the system has the opportunity to provide additional value-added services, such as print fulfillment, additional transformations (e.g., scaling, cropping, etc.), or the like. These additional value-added services would be unattractive or not feasible if instead the system were processing a decimated copy (i.e., given the level of noise introduction produced by decimations). In this manner,

any service that is preferably performed against a source object may now be easily integrated into the system, by using the ORI information provided by the present invention. For example, if a user received a decimated image on a hand-held PDA device, that user may still order a high-quality version of that image (e.g., as print fulfillment, T-shirt, coffee mug, key chain, or the like).

[0060] While the preferred embodiment has been illustrated as an add-on unit for an existing message switch, a first alternative embodiment may be created such that the media ORI module 350 includes direct connectivity to terminals or clients. In such an embodiment, therefore, the media ORI module 350 is modified to include connectivity functionality (e.g., HTTP and TCP/IP connectivity) for allowing the module to host sessions with individual terminals. In particular, the Internet may be used as an interface to bear the same application commands between terminals and the media ORI module 350, that were done between the message switch 320 and the module 350. In such an embodiment, mobile terminals may communicate with the ORI module 350 directly via the Internet, and enjoy the same benefits described above that are available as a result of being able to identify the original media source

(notwithstanding the fact that the mobile terminal may have itself only received a decimated copy). Typically, the terminals or clients would include client-side logic for facilitating communication with the module 350. The client-side logic may be pre-existing (e.g., installed driver) or downloaded dynamically (e.g., Java script, ActiveX control, or the like).

[0061] A second alternative embodiment may be created such that a portion (or all) of the functionality of the media ORI module 350 is pushed out to the individual terminals or clients. Embodiment of even a portion of the module 350 at the clients may allow some of the workload to be distributed over a number of machines, thus potentially improving throughput. Additionally, if clients are able to ascertain that media does not have ORI information, network communication for this particular process is avoided. Storage of original source media may be maintained centrally (e.g., in a central repository) or may be maintained at the respective clients (i.e., retrieved using peer-to-peer technique). These clients may participate in regular message switching systems/services, but they on their own have the ability to re-create the source (in its original fidelity). This improves the overall performance of the ex-

isting switching system, since clients then will present original source media, not decimated copies.

DETAILED OPERATION

[0062] The following description presents method steps that may be implemented using computer-executable instructions, for directing operation of a device under processor control. The computer-executable instructions may be stored on a computer-readable medium, such as CD, DVD, flash memory, or the like. The computer-executable instructions may also be stored as a set of downloadable computer-executable instructions, for example, for downloading and installation from an Internet location (e.g., Web server).

[0063] *Methodology for handling incoming new media*

[0064] Figs. 4A-B illustrate a method 400 of the present invention for handling new (original) incoming media (i.e., media without ORI information); for purposes of illustration, the method is illustrated at the point that new media items are created. At step 401, new media items are created at the client, typically a mobile (wireless) terminal. The items themselves can be a variety of types, such as images, audio, video, documents, and the like. At step

402, the client composes a message containing one or more new media items. Next, the client transmits the message to the message switch (e.g., using the MM1 protocol), as indicated at step 403. The message switch, after initial processing (e.g., user identification/authentication/billing), relays the media items to the ORI services driver or plug-in, at step 404. (Typically, the services driver would reside on the same machine as the message switch.)

[0065] Now, at step 405, the ORI services driver communicates with the media ORI module, to present the media items or objects. For this purpose, a "Present Media" application command is employed. At step 406, the media ORI module scans the media, looking for potentially existing (i.e., embedded) ORI information. Assuming no existing ORI information is found, the module assigns an object reference identifier (ORI) for the media, at step 407. The identifier may be created using a hash of the current timestamp plus an ascending counter. If desired, other attributes or components may be added to the identifier, such as an install ID (e.g., identifying the present message switch). Other potential techniques may be used to create the identifier. All that is required for the method is that

the identifier be unique (over the design life of the product for the domain of media objects reasonably expected to be encountered).

[0066] At step 408, the media ORI module embeds the ORI in the media item. In the currently preferred embodiment, the ORI is embedded as a binary string (e.g., 256-bit binary string). A number of techniques exist for embedding information within media objects or items. For example, most standard media types include a header that permits storage of arbitrary data, such as storage within a header of an image file or a control channel of a video (e.g., MPEG-4) file. Alternatively, one could use a watermark approach to interleave information with the underlying media data itself (with human-imperceptible loss of quality). Similarly, newer media types include digital rights management (DRM) features that allow additional information, for example, to be added or piggybacked onto a DRM subchannel. Although a variety of techniques are potentially available for embedding information, the particular technique chosen for a given media type should be one that allows the embedded information to be resilient/redundant to decimation. If desired, multiple techniques may be used on a single media object, for example, in an

effort to further guard against destruction by transformation.

[0067] All told, redundant mechanisms based on existing media encoding may be used to embed ORI information in a manner that will survive decimation of a given a media type. It is worth noting that although an embedded ORI identifier may not be guaranteed to survive all possible decimations, the identifier typically will not interfere with media rendering/processing and thus will not serve as a point of failure. In other words, although encoding of certain media types may create difficulty for embedding the identifier, the embedding itself creates no downside since it will not lead to failure of rendering/processing. Therefore, the ORI may be lost in certain failure modes, but in such a case the media object simply reverts back to an ordinary media object (i.e., one not embellished with ORI information).

[0068] At step 409, the embellished media item, which now contains an embedded original reference identifier (ORI), is stored in the original media repository (and it may optionally be added to an index for faster future retrieval). Finally, at step 410, the embellished media item is returned via the application protocol/interface as ORI-enhanced

media to the message switch, which may now proceed to process the media item in a normal manner (e.g., transform and deliver it in a multimedia message destined for particular target/destination terminal(s)). Since the ORI survives decimation, further use of the media item may be done in a manner that allows re-association with the original source media (as will next be described).

[0069] *Methodology for handling incoming known (ORI-embellished) media*

[0070] Figs. 5A-B comprise a flowchart of a method 500 for processing incoming media objects that contain ORI information. Several of the initial steps are the same as the method 400. In particular, steps 401-405 are essentially repeated for the method 500, except that the incoming media object is now one that does contain ORI information. Therefore, steps 501-505 are shown essentially unchanged from the method 400. At the sixth step (corresponding to step 406), existing ORI information is found during scanning. This is indicated at step 506. At step 507, the media ORI module retrieves the original media from the repository. Optional indexing techniques are used to speed retrieval, if desired. Additionally, a cache could be employed for holding copies of frequently-ac-

cessed media objects. Now, at step 508, the original-quality media is returned back to the message switch -- that is, to serve as substitute media for the decimated media that was presented (at step 501). The returned media of course includes embedded ORI information so that subsequent use of the media may again be re-associated with the original source. If desired, the return of original-quality media may be contingent upon the user's rights (i.e., privilege level), for example, based on DRM information contained within the presented media.

[0071] *Embedding of original reference identifier (ORI) in media*

[0072] As previously described, the original reference identifier (ORI) may be embedded in media. Many image formats, especially JPEG, provide the ability to store textual and binary thumbnail metadata within the file itself. Digital cameras, scanners and other imaging equipment will often generate technical data and store it in an image in a more or less standard format known as the Exchangeable Image File Format or EXIF.

[0073] In one embodiment of the present invention, the ORI is embedded in a JPEG media object. Fig. 6 is a simplified block diagram of the JPEG image file format. Although there are additional headers and marker fields described

in the standard specification for JPEG, they are considered optional, and in the interests of clarity of description, are omitted from the figure. The fields are as follows.

[0074] Image Start Marker (601): A two-byte sequence that denotes the start of a JPEG image (0xFF 0xD8)

[0075] Image End Marker (603): Similar to the Image Start Marker, but found at the end (0xFF 0xD9)

[0076] Frame (602): The Frame represents the beginning of actual image data. A JPEG file can contain at most one Frame.

[0077] Misc/Tables Section (611): The JPEG format allows for arbitrary amounts of information to be embedded in the headers of an image, using application-specific encodings. The format itself stores some necessary information in this section, including the quantization matrix, and Huffman codes.

[0078] Frame Header (612): The frame header includes important information about the structure of the frame that follows, including the number of lines, the number of samples in each line, and the list of components in the image.

[0079] Scan (613): A scan represents a single pass, or a single component of the image. For example, a black and white image could be encoded in a single scan, while a color

image would typically take multiple scans.

[0080] Scan Header (622): The Scan Header contains information necessary to decode the data (like which Quantization table to use), and information necessary to put that data in the right place once it has been decoded (like x and y scaling factors).

[0081] Encoded Data (623): Encoded Data is the output of the JPEG encoding process.

[0082] Of particular interest herein is the Misc/Tables Section 611, which allows for arbitrary amounts of information to be embedded in the headers of an image, using application-specific encodings. The section may be used to accommodate application-specific encoding for the purpose of embedding or hiding ORI information without visibly affecting the image quality and without being easily removed during transformation. For a JPEG media object, the ORI (object reference identifier) embedding unit 361 may include program logic for identifying the JPEG media type by using the two-byte sequence that denotes the start of a JPEG image (0xFF 0xD8). The ORI assign unit 363 includes program logic for assigning a new ORI value for an incoming media object that has never been seen before and places that ORI value (e.g., as a binary string)

in the Section 611, using application-specific encodings. In this manner, arbitrary binary data may be embedded within the media object, thus allowing each decimated copy to retain enough information (via the ORI) to allow the copy to be re-associated with the original media object.

[0083] In another embodiment, a digital watermark technique may be employed. Content providers have a need to embed signals into video/multimedia data, including audio data, which can subsequently be detected by software and/or hardware devices for purposes of authenticating copyright ownership, controlling copying and display, and ownership management. The watermark itself is an identification code that is embedded in the original digital data and is preferably imperceptible to the human observer of the artistic work. One example of a scheme for watermarking involves inserting an identification string into a digital audio signal to substitute the insignificant bits of randomly selected audio samples with the bits of an identification code. Another example of watermarking relating to watermarking video digital works involves assigning a predetermined value to a predetermined coding parameter that, when modified, requires further parameters to be

modified in order to correctly decode the video signal. In one watermarking technique, each copy of an object is marked with an identifier code.

[0084] Digital watermarks are typically used to mark each individual copy of a digitized work with information identifying the title, copyright holder, and even the licensed owner of a particular copy. Watermarks can also serve to allow for secured metering and support of other distribution systems of given media content and relevant information associated with them, including addresses, protocols, billing, pricing, or distribution path parameters, among the many things that could constitute a "watermark." For example, U.S. Patent No. 5,905,800 to Moskowitz et al. describes a method for applying a digital watermark to a content signal. As a another example, U.S. Patent No. 5,991,426 to Cox et al. describes a technique where a digital watermark is inserted into multimedia data. As yet another example, U.S. Patent No. 6,611,599 to Natarajan describes a watermarking technique that includes encrypting a message derived from source data on the digital object. The foregoing patent references are hereby incorporated by reference.

[0085] For those media objects already employing digital water-

marks (suitable for surviving transformations), the ORI information may be incorporated into the watermarks themselves, thus eliminating the need to encode the ORI information in a separate field (e.g., header field) of the media object. In accordance with the present invention, the ORI (object reference identifier) embedding unit 361 may include program logic for identifying a particular watermark type by using the watermark's signature type (e.g., byte sequence). The ORI assign unit 363 includes program logic for assigning a new ORI value for an incoming media object that has never been seen before and places that ORI value (e.g., as a binary string) in the watermark, using encoding appropriate for the particular watermark. As with the JPEG embodiment, each decimated copy can retain enough information via the ORI to allow the copy to be re-associated with the original media object. It should be appreciated by those skilled in the art that the methodology described herein for use with digital watermarks is not tied to a particular watermarking technique but, instead, is suitable for use with any digital watermark capable of surviving transformations/decimation to a point that would allow extraction or reconstitution of the ORI information.

[0086] While the invention is described in some detail with specific reference to a single-preferred embodiment and certain alternatives, there is no intent to limit the invention to that particular embodiment or those specific alternatives. For instance, those skilled in the art will appreciate that modifications may be made to the preferred embodiment without departing from the teachings of the present invention.